Adaptive P2P Video Streaming Via Packet Labeling

Jacob Chakareski[a] and Pascal Frossard[b]

[a]Layered Media
Rochelle Park, NJ 07662, USA
[b]Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

ABSTRACT

We consider the scenario of video streaming in peer-to-peer networks. A single media server delivers the video content to a large number of peer hosts by taking advantage of their forwarding capabilities. We propose a scheme that enables the peers to efficiently distribute the media stream among them. Each of the peers connects to the streaming server via multiple multicast trees that provide for robustness in the event of peer disconnection. Moreover, adaptive forwarding of the media content at each peer is enabled by labeling the packets with their importance for the reconstruction of the media stream. We study the performance of the proposed scheme as a function of system parameters such as the play-out delay of the media application, the peer population size and the number of multicast trees employed by the scheme. We show that by placing priorities on forwarding the individual packets at each peer an improved performance is achieved over conventional peer-to-peer systems where no such prioritization is deployed. The gains in performance are particularly significant for low-delay applications and large peer populations.

Keywords: peer-to-peer, packet scheduling, rate-distortion optimized streaming

1. INTRODUCTION

Peer-to-peer (P2P) overlay networks have emerged as the most dominant source of traffic in the Internet today.[1] Their inherent bandwidth scalability and robustness to single point failures have made them very attractive for content sharing applications among large populations of peer host computers on the Internet.[2] On the other hand, their dynamic nature caused by peers connecting and disconnecting at will, coupled with the typically limited peer forwarding capacity and ineffective packet routing, have proved to be their major shortcomings towards P2P streaming applications, up to date. In particular, these disadvantages contribute to exceedingly long latencies of media packets which contradict with their characteristically strict timing constraints as imposed by their associated streaming application.

Still, researchers have considered P2P networks as an attractive lower (than conventional server-client architectures) cost solution for streaming media to large audiences as well as for handling flash crowds which can collapse conventional media delivery networks. In particular, prior work on P2P streaming addresses its drawbacks in several ways. The work of Padmanabhan et al.[3] attempts to provide robustness to peers' departures and to reduce latencies by employing multiple description encoded media content that is distributed to clients over multiple independent multicast trees. The work in[4] employs SP and SI frames to adaptively stop error propagation due to packet loss and distortion-optimized retransmission requests to recover the most important missing packets at each peer. The authors in[5] consider a somewhat different approach. They propose to combine a conventional server-client architecture with peer-to-peer connections among clients in order to provide improved bandwidth scalability and reduced client waiting times for video-on-demand applications. Nonetheless, despite proving feasibility initial implementations of actual peer-to-peer streaming systems on the Internet[6,7] still exhibit unacceptable start-up latencies and unstable video playback.

In this paper, we propose a framework for peer-to-peer adaptive streaming based on multiple multicast trees and packet prioritization via labeling. Specifically, each peer is connected to multiple distribution trees that deliver the media content from a streaming server to a client (peer) population. Moreover, each packet is characterized ahead of time by the streaming server with a label that enables prioritized packet forwarding at

each peer. This enables enhanced performance over systems where no such prioritization is deployed, especially in the case of low play-out delays and large number of peers.

The rest of the paper is structured as follows. First, in Section 2, we describe the control protocols employed in our framework for handling the media distribution via multiple multicast trees. In particular, we describe how new peers join the distribution and how the system deals with peer departures. Then, in Section 3, we explain the algorithm for prioritized packet forwarding at each peer based on a priori labeling of the packets with their respective importance for the reconstruction quality of the media stream. Next, in Section 4, we examine the performance of the proposed peer-to-peer streaming framework via simulation experiments. Finally, the paper ends with concluding remarks provided in Section 5.

2. PEER-TO-PEER MEDIA DELIVERY VIA MULTIPLE MULTICAST TREES

In our framework, the media content is distributed from the original streaming server to the peer population via application-layer multicast.[8] Multiple multicast trees are dynamically established and maintained over the overlay (P2P) network so that the media content is forwarded along these trees simultaneously. The server sends complimentary packet partitions of the overall stream over each of the trees, as in.[4] Therefore, a peer needs to be connected to all trees at the same time in order to receive the complete set of media packets. The media server is the root of all trees.

We decided to employ this approach to distribute the media content across the multiple trees as it is the least bandwidth demanding. Alternative approaches comprise sending redundancy packets in the form of forward error correction as in[9] and multiple descriptions as in,[3] and are more bandwidth demanding for delivering the same media stream. However, it should be noted that these latter approaches handle the effects of missing packets* at a peer without retransmissions, which in turn are employed in our system for the same purpose, thereby causing extra latencies. Still, as our experimental findings show these extra latencies do not penalize the performance of the proposed system noticeably due to the prioritized packet forwarding employed at each peer.

The building and the maintaining of the multicast trees is performed in a distributed fashion by the peers, save for the fact that the streaming server (the root node) maintains a partial list of connected peers that is used to allow new nodes to join the system and to handle repairing broken trees upon random node disconnections, i.e., departures from the system. In the next section, we describe in details the protocols that are employed in our framework for handling these two events. Note that our approach is different from prior works such as[10] that employ centralized tree management typically performed by the root node.

2.1. Joining the multicast

When a peer decides to join the multicast session, it contacts the streaming server first asking for prospective nodes through which the peer can connect to the session. Then, the peer contacts each of the nodes from the partial list that the server provides and waits for their response. The communication with the server and the prospective nodes is done by the peer via a handshaking procedure, involving message exchanges. The nodes respond to the peer by providing their current available forwarding bandwidth† and the number of hops through which they are connected to the server in every tree.

Once the peer hears from the nodes, it proceeds to select an appropriate parent in every multicast tree by employing the following two criteria. First, only those nodes that have sufficient forwarding bandwidth left (at least equal to the average date of the requested video stream) to support the peer on a tree are selected from the partial list provided by the server. Then, among these candidate nodes a winner is selected according to the number of hops through which the candidates are connected to the root node in that tree. In particular, the parent node with the smallest number of hops is chosen in order to minimize the latency in delivering the media packets from the server to the joining peer on this tree. The protocol resolves prospective ties by selecting a different‡ parent node for the current tree under consideration. This increases the robustness of the system to node departures by maintaining some sort of delivery diversity.

*These are caused by temporary disruptions of some of the multicast trees due to random peer departures.

†We assume that the nodes can accurately measure their upload bandwidth.

‡From the parents and the respective trees already selected for the peer.

A node sends a small control packet periodically upward every tree informing its parents of its presence. In turn, a parent confirms its presence by immediately responding to such control messages. In this way, each node can obtain the topology of the subtrees to which it is connected. This information is then provided to newly connecting peers looking for appropriate parent nodes in every tree, as described above. Furthermore, these lists of ancestors in every tree that a node maintains can also help to deal with tree disruptions due to node departures, as explained next.

2.2. Handling node departures

Upon disconnection, a peer stops forwarding media packets to the subtrees for which it acted as a parent. When the children nodes on these subtrees detect that the flow of packets to them has been stopped, they attempt to probe the disconnected peer, to which this parent is unresponsive. Therefore, the children nodes attempt to rejoin the broken tree by contacting other nodes further upward (from the disconnected parent) in the same tree. If this fast rejoin procedure is unsuccessful, the children nodes connect the streaming server again and ask for an updated partial list of connected peers. Then, each of these nodes repeats the procedure described in Section 2.1 to join again the same multicast tree.

Now, due to the temporary tree disruption some media packets will be missing at the children nodes. Hence, while reconnecting these nodes also send send retransmission requests via the other existing trees to upstream parents that subsequently forward the missing packets. The retransmission requests are sent via small control packets. Note that these are application layer retransmission requests that are sent only selectively. Therefore, they do not have the same feedback implosion effect as in network layer multicast. Furthermore, the effects of extra latency caused by these retransmissions are mitigated by the prioritized packet forwarding employed at each node, as explained earlier.

3. PRIORITIZED FORWARDING VIA PACKET LABELING

Each peer schedules the transmission of packets on its uplink according to the algorithm described here. There are three parts to the scheduling procedure: (1) labeling the packets with their respective importance at the streaming server, (2) computing priorities for forwarding packets at a peer based on these importance labels, and finally (3) scheduling the actual packet transmissions according to these priorities. In the following, we describe each of these constituent components in greater detail.

3.1. Characterizing media packets

A packet is characterized with its importance for the overall reconstruction quality of the media stream a priori by the streaming server. The value of the packet's importance λ is computed using the algorithm described in[11] and the server labels the packet with this quantity before forwarding it along one of the multicast trees. The importance λ of a packet can simply be encoded by the media server in the packet's header.

3.2. Computing packet priorities

Let there be N packets[§] in the outgoing queue at a peer (denoted as p) that need to be forwarded to the children nodes of this peer at present. Let $\lambda(i)$ denote the importance of packet i in the queue, for $i = 1, \ldots, N$, let m be one of the children nodes of the peer to which packet i needs to be forwarded, and finally, let $Tree(i, m)$ be the corresponding subtree sizefor packet i at node m. That is, $Tree(i, m)$ denotes the number of nodes in the subtree (whose root is m) to which packet i should be delivered as distributed along the corresponding multicast tree[¶].

Then, let $P(i, m)$ denote the priority of forwarding packet i to node m that is computed as follows

$$P(i,m) = \lambda(i) \frac{Tree(i,m)}{\sum_{i=1}^{N} \sum_{k \in C(i,p)} Tree(i,k)}, \tag{1}$$

[§]Including those that have been requested for retransmission.

[¶]m computes this information by counting the number of control packets it receives from its descendant nodes in the subtree, by which the nodes announce their presence, as explained in Section 2.1.

where $C(i,p)$ denotes the set of children nodes to which peer p should forward packet i on its multicast tree. Note that such computed priority for a media packet not only takes into account the importance of the packet for the overall quality of the reconstructed media stream at a receiver, but also recognizes the size of the peer population that will be affected if the packet is not delivered on time.

3.3. Scheduling transmissions

A peer p sorts in decreasing order the priorities $P(i,m)$ associated with the packets in its queue. Then, whenever the peer is ready to forward the next packet, it sends on the uplink the packet corresponding to the head of the sorted list and the list of sorted priorities is updated afterwards by removing this packet from the head of the list. Similarly, the list is updated by the peer every time a new packet arrives that needs to be forwarded or a new retransmission request is received from another peer. This is done by appropriately inserting the priority associated with the new packet (or the new retransmission request) in the sorted list. In order to minimize the effects of congestion on the uplink, a new transmission on this link occurs only after waiting for the period of time that is necessary to forward the previously sent packet on the uplink including some extra time to account for prospective transmissions of control packets in the meantime.

4. SIMULATION EXPERIMENTS

4.1. Setup

In this section, we evaluate the performance of the proposed P2P streaming system. The media content employed in the experiments is the test video sequence Foreman in CIF frame size encoded at 30 fps using an H.264 codec.[12] The sequence is encoded with a constant quantization level at an average luminance (Y) PSNR of about 35.5 dB and a Group of Pictures (GOP) size of 32 frames, where each GOP comprises an I-frame followed by a repeating pattern of three B-frames and a P-frame, i.e., IBBBPBBBP.... The average encoding rate of the video content is 330 kbps.

The compressed video frames are packetized into UDP packets at transmission. Those frames that exceed the size of the maximum transmission unit are fragmented into multiple packets before transmission. The streaming server forwards successive packets over the multiple multicast tree in a round-robin fashion. Video quality at a peer is measured in terms of the average Y-PSNR of the reconstructed video frames at the peer. Missing frames (including those that arrive late) are replaced by the peer using previous frame error concealment.

In the simulations, we examine two values for the number of peers participating in a session: 50 and 250. The distribution of the uplink and downlink bandwidth at the peers is selected according to the findings reported in[13] and corresponds to the popularity of access link technologies encountered in practice today. Peers join and depart the multicast session at random, where the average time that a peer spends in the session is set to 7 minutes. The transmission delay on each network link is 5 ms and the maximum number of hops between a peer and the streaming server is 9 hops.

4.2. Performance evaluation

In the first experiment, we examine the average video quality over 250 peers as a function of the number of multicast trees employed for streaming the media content. These results are shown in Figure 1. Two things can be summarized from Figure 1. First, the average video quality tremendously improves when the content is distributed over more than one multicast tree. When only one tree is employed for multicasting, more than 50% of the participating peers act as free-riders, that is, they can only receive media content but cannot forward it. That is because their uplink capacity is smaller than the data rate of the media stream. Hence, the poor video quality for streaming over a single multicast tree. On the other hand, when more than one tree is employed there are no more free riders and every peer can contribute with its forwarding capacity on the uplink. In particular, now the data rates of the stream partitions sent along each tree are smaller than the forwarding capacity of any peer. Hence, the substantial improvement in performance relative to the single tree case.

The second observation stemming from Figure 1 is that steady video quality is maintained, when two or more multicast trees are employed for streaming. This is happening despite the fact that the amount of control traffic sent among the peers increases linearly with the number of multicast trees employed in a session. Specifically, as

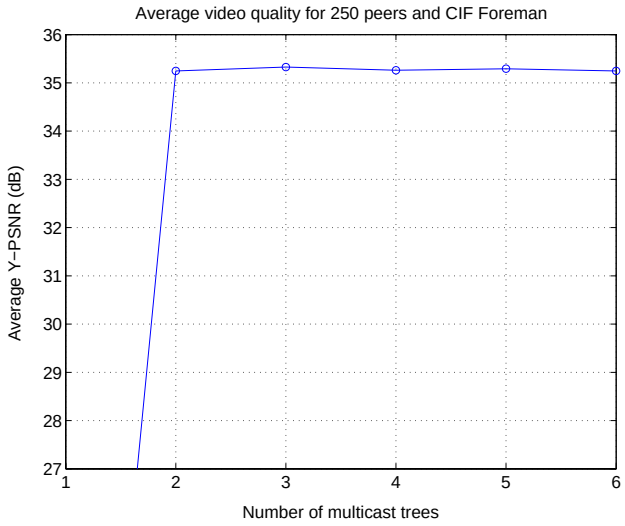

Figure 1. Average Y-PSNR (dB) vs. Number of multicast trees for CIF Foreman and 250 peers.

the size of control traffic on the network increases, the amount of free network bandwidth that can be employed for streaming reduces. However, the reduction in available bandwidth is accounted for by employing prioritized forwarding and retransmission of packets, as explained in Section 3. Moreover, these two also compensate for the effects of increased frequency of peer departures caused by the increase in number of multicast trees. Hence, there is no noticeable reduction in video quality as the number of multicast trees increases beyond two, as seen from Figure 1.

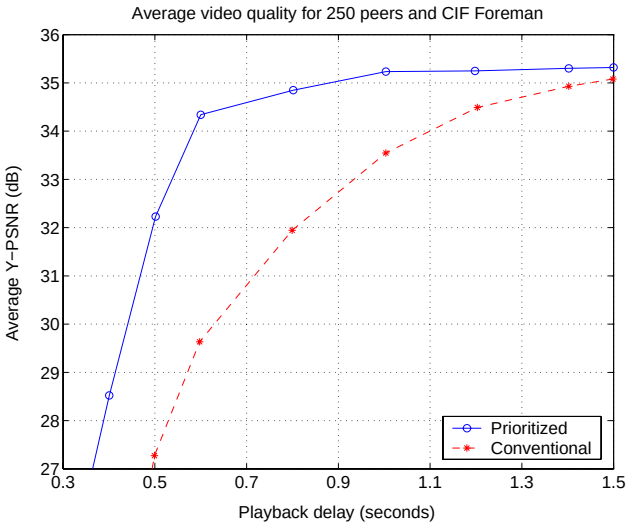

Figure 2. Average Y-PSNR (dB) vs. Playback delay for CIF Foreman and 250 peers.

In the second set of experiments, we investigate the impact of prioritized forwarding as employed in our P2P streaming system. In particular, we study the performance of the proposed system as a function of the playback delay of the media application running at each peer. This is the time difference between the instance when a media data unit is generated by an encoder and the time when the data unit needs to be decoded and (dis)played

at a receiving peer. In the simulations, we also evaluate the performance of a conventional P2P system where no prioritized forwarding is deployed. Specifically, in this system, a node forwards media packets in arriving order, only dropping those packets that have already expired by the time they have arrived at the node.

In Figure 2, we examine the performances of the two P2P systems, with prioritized forwarding and conventional, when 250 peers are participating in a session. It can be seen that for long playback delays, the two systems perform alike. On the other hand, as the playback delay is reduced[‖], the difference in performance between conventional and prioritized forwarding systems becomes quite substantial. Alternatively, by employing adaptive and prioritized packet forwarding at each peer, the proposed system is able to provide an up to 50% reduction (relative to the conventional system) in start-up latency required for a given video quality to be achieved, as shown in Figure 2.

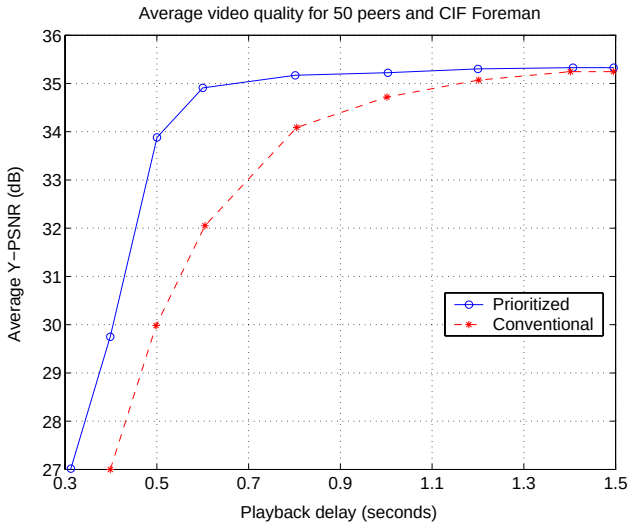

Figure 3. Average Y-PSNR (dB) vs. Playback delay for CIF Foreman and 50 peers.

Finally, it can be seen from Figure 3 that the performance improvement of the proposed system becomes less significant, when there are fewer peers participating in a session. This is expected, because for a peer population of size 50, there is less forwarding that needs to be done over the multicast trees, in general. Hence, the resulting latencies in delivering the media packets are smaller now, which makes prioritized packet forwarding less important for the overall performance.

5. CONCLUSIONS

We have presented a system for peer-to-peer streaming based on multiple multicast trees and prioritized packet forwarding and retransmission at every peer. Via simulations we have shown that it is important to employ multiple multicast trees in order to take advantage of the forwarding capacity of every peer. Moreover, we have also shown that prioritized packet forwarding can provide substantial gains in performance in the case of large peer populations and low latency applications.

REFERENCES

1. CacheLogic, LTD., "Internet traffic reports," http://www.cachelogic.com.
2. Sharman Networks, LTD., "KaZaA," http://www.kazaa.com/.
3. V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-2002-37, Apr. 2002.

[‖]This corresponds to media applications with stringent latency requirements.

4. E. Setton, J. Noh, and B. Girod, "Rate-distortion optimized video peer-to-peer multicast streaming," in *Proc. ACM Workshop on advances in peer-to-peer multimedia streaming*, Singapore, Singapore, Nov. 2005, pp. 39–48.
5. C. Dana, D. Li, D. Harrison, and C.-N. Chuah, "BASS: BitTorrent assisted streaming system for video-on-demand," in *Proc. Workshop on Multimedia Signal Processing.* Shanghai, China: IEEE, Oct./Nov. 2005.
6. PPLive, http://www.pplive.com/.
7. X. Zhang, J. Liu, and B. Li, "On large-scale peer-to-peer live video distribution: Coolstreaming and its preliminary experimental results," in *Proc. Workshop on Multimedia Signal Processing.* Shanghai, China: IEEE, Oct./Nov. 2005.
8. S. Paul, *Multicasting on the Internet and its Applications.* Kluwer, 1998.
9. T.-W. A. Lee, S.-H. G. Chan, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang, "Allocation of layer bandwidths and FECs for video multicast over wired and wireless networks," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1059–1070, Dec. 2002.
10. V. Padmanabhan, H. Wang, and P. Chou, "Resilient peer-to-peer streaming," in *Proc. Int'l Conf. on Network Protocols.* Atlanta, GA, USA: IEEE, Nov. 2003, pp. 16–17.
11. J. Chakareski, "Distributed media cooperation for enhanced video communication," in *Proc. Int'l Packet Video Workshop*, Hangzhou, China, Apr. 2006, pp. 773–783.
12. Telecom. Standardization Sector of ITU, "Video coding for low bitrate communication," *Draft ITU-T Recommendation H.264*, Mar. 2003.
13. K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting largescale live streaming applications with dynamic application endpoints," in *Proc. Data Communication, Ann. Conf. Series.* Portland, OR, USA: ACM, Aug. 2004.